DNA Nanotechnology
Nano Tasks

- **Sensing**
  - Binding to specific molecules

- **Computing**
  - Analog: Signal Filtering or Amplification
  - Digital: Logical gates

- **Actuating**
  - Releasing molecules
  - Producing forces

- **Constructing**
  - By self-assembly
  - Or under 'program' control

- **Nucleic Acids (DNA/RNA)**
  - Probably the only materials that can perform all these functions.
  - Technology relatively well developed.
  - Can interface to biological entities.

---

Aptamers (Sensors)

Fig.: RNA-aptamer - purine complex
POD-ID: 1015 generated by VMD 1.8.2
Computation: Curing Cancer with one AND Gate

**letters to nature**

An autonomous molecular computer for logical control of gene expression

Yasuko Nomura,1‡, Hayamizu GY,‡, Eh Reen-Dar,1 Finka Ader,1 & Eshed Shapiro2

1Department of Computer Science and Applied Mathematics and 2Department of Biological Chemistry, Weizmann Institute of Science, Rehovot 76100, Israel

---

**Diagram Description**

**a. Computation module: logical analysis of disease indicators**

- **Yes, PPAP2B↓** → Yes
- **Yes, GSTP1↓** → Yes
- **No, PIM1↑** → No
- **Yes, HPN↑** → Yes
- **No, Yes** → Positive diagnosis
- **No, No** → Negative diagnosis

**b. Input module: software regulation by mRNA levels**

- **Active Yes, PM1↑** → Yes
- **Inactive Yes, PM1↓** → Yes

**c. Probabilistic check for PM1↑ indicator**

- **High active Yes, PM1↑** → Yes
- **Low inactive Yes, PM1↓** → No

**d. Output module: drug administration**

- **MDM2 mRNA**
  - **High administered drug** → Yes
  - **Low suppressed drug** → No
- **MDM2 protein**
Actuators

DNA Tweezers
(Yurke & Turberfield, Nature 2000)

"The fuel strand attaches to the handles and draws the two arms of the tweezers together."

DNA Walkers
(Yin, Choi, Calvert & Pierce, Nature 2008)
Compositionality

- **Sensors and Actuators at the 'edge' of the system**
  - They can use disparate kinds of inputs (sensors) and outputs (actuators)

- **The 'kernel' of the system computes**
  - Must use uniform inputs and outputs

- **Compositionality in the kernel**
  - Supporting 'arbitrary' computing complexity
  - The output of each computing component must be the same kind of 'signal' as the input
    - If the inputs are voltages, the outputs must be voltages
    - If the inputs are proteins, the outputs must be proteins
    - If the outputs are photons the inputs must be photons
    - If the inputs are DNA, the outputs must be DNA

- What should our nano-signals be?
what does dna compute?

● electronics has electrons
  o all electrons are the same
  o all you can do is see if you have few (‘false’) or lots (‘true’) of electrons
  o hence boolean logic is at the basis of digital circuit design
  o symbolic and numeric computation has to be encoded above that
  o but mostly we want to compute with symbols and numbers, not with booleans

● dna computing has symbols (dna words)
  o dna words are not all the same
  o symbolic computation can be done directly
  o we can also directly use molecular concurrency

● process algebra as the ‘boolean algebra’ of dna computing
  o what are the ‘gates’ of symbolic concurrent computation?
  o that’s what process algebra is about
  o (process algebra comes from the theory of concurrent systems)
Implementing "Arbitrary" Computing Functions
Molecules as Automata (DNA14 Invited Talk)

Continuous-state Semantics (Mass Action Kinetics)

ODE = ODE

Continuous Chemistry

Discrete Chemistry

Discrete-state Semantics (Chemical Master Equation)

CTMC = CTMC

Process Algebra

The Real Wet Stuff

D. Soloveichik, G. Seelig, E. Winfree. DNA as a Universal Substrate for Chemical Kinetics. Proc. DNA14.

L. Cardelli: “On Process Rate Semantics” (TCS)

L. Cardelli: “A Process Algebra Master Equation” (QEST’07)
DNA Compilation

Separating Circuit Design from Gate Design

Circuit Design Space

- Discrete Chemistry
- Interacting Automata
- Boolean Networks
- Finite State Automata
- Petri Nets

High level languages (TBD)

Low level languages

Sequence Design

DNA

DNA Sequence Design

Circuit Design Space
DNA Compilation

Separating Circuit Design from Gate Design

High level languages (TBD)

Low level languages

Circuit Design
(e.g. half-adders from Boolean gates)

Gate Design Space

DNA

Sequence Design

Higher-level languages

Discrete Chemistry

Interacting Automata

Boolean Networks

Finite State Automata

Petri Nets

Strand Algebra

Seesaw Gates
DNA Compilation

Separating Circuit Design from Gate Design

- Higher-level languages
- Discrete Chemistry
- Interacting Automata
- Boolean Networks
- Finite State Automata
- Petri Nets

DNA gate implementation

- Strand Algebra
- Strand Displacement
- Verification of DNA gate implementation

Circuit Design
(e.g. half-adders from Boolean gates)

Gate Design
(e.g. Boolean gates from transistors)

Sequence Design

Cardelli and Phillips, A Programming Language for Composable DNA Circuits, Royal Society Interface Journal
DNA Compilation

Separating Circuit Design from Gate Design

High level languages (TBD)

Low level languages

Circuit Design
(e.g. half-adders from Boolean gates)

Gate Design
(e.g. Boolean gates from transistors)

DNA

Seesaw Gates

Interacting Automata

Discrete Chemistry

Boolean Networks

Finite State Automata

Petri Nets

Higher-level languages

Sequel

Other DNA Mechanisms

Strand Displacement

Other DNA Mechanisms

DNA Sequence Design

Luca Cardelli 2009-07-11

2009-07-11 Luca Cardelli

14
DNA Compilation

Separating Circuit Design from Gate Design

High level languages (TBD)

Low level languages

Discrete Chemistry

Interacting Automata

Boolean Networks

Finite State Automata

Petri Nets

Seeing Gates

Strand Algebra

Other DNA Mechanisms

Strand Displacement

Other DNA Mechanisms

Rest of the talk: bottom up
Toehold Mediated Strand Displacement
**Watson-Crick Duality**

- **G - C**
  - **T - A**
  - **Affinity**

- **G⊥ = C**
  - **T⊥ = A**
  - **Complementarity**

- **G: A: C: T**
  - **C: T: G: A**
  - **Double Strand**

- **Equal Single Strands**

- **Complementary Single Strands**

- **Hence (G: A: C: T)⊥ = A: G: T: C = T⊥: C⊥: A⊥: G⊥**

- **(X: Y)⊥ = Y⊥: X⊥**
  - **Watson-Crick duality**
  - (for any sequences of bases X, Y)
Hybridization

a, b, c, etc. denote DNA (sub)sequences with Watson-Crick complements $a^\perp, b^\perp, c^\perp$, etc.

Hybridization is also called annealing; denaturation is also called melting.

The direction of the reaction (or in general the equilibrium between the two states) is determined by a number of factors, e.g. temperature.

We assume we are in conditions that favor hybridization beyond a certain length of matching region.
Gate Elements: Short and Long DNA Segments

Short (red) segments

Long (black) segments

Reversible Binding

Irreversible Binding
Gate Elements: **Basic Mechanisms**

**Irreversible**

**Strand Displacement**

**Reversible**

**Toehold Exchange**
**Gate Elements: Signals and Gates**

- Signals “x” are single-stranded and ‘positive’
  \[ x_h = \text{history} \quad x_t, x_b = \text{signal identity for } x \]
  \[ x_t = \text{toehold} \]
  \[ x_b = \text{binding} \]

- This 3-segment signal representation is original to this work, it is based on the 4-segment signals of D. Soloveichik, G. Seelig, E. Winfree. Proc. DNA14, but leads to simpler and more regular gate structures.

- Gate backbones are double-stranded, except for ‘negative’ toeholds.

- Separation of strands and gates helps the DNA realization, as one can use 3-letter alphabets (ATC/ATG) for each strand, minimizing secondary structure and entanglement.
Circuit Elements: **x.y Transducer Gate**

\[ x \mid x.y \rightarrow y \]

\[ G_b, G_t \text{ (gate backbone and trigger) form the transducer.} \]

Any history segment that is not determined by the gate structure is said to be ‘generic’ (can be anything).

Any gate segment that is not a non-history segment of an input or output signal is taken to be ‘fresh’ (globally unique for the gate), to avoid possible interferences.
A Fork signal-processing gate takes a signal $x$ and produces two signals $y,z$ according to the reaction $x \mid x. [y,z] \rightarrow y \mid z$.

$G_b, G_t$ (gate backbone and trigger) form the gate.

Any history segment that is not determined by the gate structure is said to be ‘generic’ (can be anything).

Any gate segment that is not a non-history segment of an input or output signal is taken to be ‘fresh’ (globally unique for the gate), to avoid possible interferences.
A Join signal-processing gate takes both signals $x, y$ and produces a signal $z$ according to the reaction $x \mid y \mid [x,y].z \rightarrow z$

The garbage $r_1$ and $r_2$ must be collected (after the gate has fired) to avoid accumulation. This can be achieved by a similar scheme taking $r_1,r_2$ as input signals.
Garbage collection of $r_1$ is needed for join to work well. This is done by another reversible-AND between $r_1$ and $r_2$, triggered by the release of $r_2$. This second reversible-AND leaves garbage too ($r_3, r_4$), but this can be collected immediately, as we know by construction that both inputs $r_1, r_2$ are available and we need not wait to revert their bindings.

The extra intermediate $c,d$ segments separate the $r_1$ binding from the $r_2$ binding. Without them, a segment $y_t : y_b$ (instead of $y_t : c$ and $d : y_b$) would be released: that is $y$!
General Join/Fork Gate

\[[x_1, \ldots, x_n], [y_1, \ldots, y_m]\]

\(x_1 \mid \ldots \mid x_n \mid [x_1, \ldots, x_n], [y_1, \ldots, y_m] \rightarrow y_1 \mid \ldots \mid y_m\)

Garbage collection
Strand Algebra

Diagram showing relationships between discrete chemistry, interacting automata, strand algebra, strand displacement, and other mechanisms.
Strand Algebra

$$\begin{align*}
P ::= & \ x : [x_1, \ldots, x_n].[y_1, \ldots, y_m] : 0 : P | P : P^* \\
& n \geq 1, m \geq 0
\end{align*}$$

- $x$ is a **signal**
- $[x_1, \ldots, x_n].[y_1, \ldots, y_m]$ is a **gate**
- $0$ is an **inert solution**
- $P | P$ is **parallel composition** of signals and gates
- $P^*$ is a **population** (multiset) of signals and gates

**Reaction Rule**

$$x_1 \ | \ .. \ | \ x_n \ | \ [x_1, \ldots, x_n].[y_1, \ldots, y_m] \rightarrow y_1 \ | \ .. \ | \ y_m$$

**Auxiliary rules** (axioms of diluted well-mixed solutions)

$$P \rightarrow P' \ \Rightarrow \ \ P | P'' \rightarrow P' | P'' \ \ \ \text{Dilution}$$

$$P \equiv P_1, P_1 \rightarrow P_2, P_2 \equiv P' \ \Rightarrow \ P \rightarrow P' \ \ \ \text{Well Mixing}$$

Where $\equiv$ is a congruence relation (syntactical ‘chemical mixing’) with $P^* \equiv P | P^*$ for unbounded populations.
Compiling Strand Algebra to DNA

\[ P ::= x : [x_1, \ldots, x_n].[y_1, \ldots, y_m] :: 0 :: P | P :: P^* \quad n \geq 1, \quad m \geq 0 \]

- **compile**(x) =

- **compile**([x_1, \ldots, x_n].[y_1, \ldots, y_m]) =

- **compile**(0) = empty solution

- **compile**(P | P') = mix(compile(P), compile(P'))

- **compile**(P*) = population(compile(P))
More in the Paper

- **Stochastic strand algebra**
  - Matches the stochastic semantics of interacting automata
  - Uses a technique for implementing constant buffered populations, to replace $P^*$ with finite populations

- **Nested strand algebra**
  - An higher-level language (with nested expressions)
  - A compilation algorithm into the basic strand algebra
Computational Abstractions ("Low-Level" Languages)
This encoding is *compositional*, and can encode *any* Boolean network:
- multi-stage networks can be assembled (*combinatorial logic*)
- network loops are allowed (*sequential logic*)
Petri Nets

Petri Nets to Strand Algebra

Transitions as Gates
Place markings as Signals

\[ ([p_1, p_2] \cdot [p_3, p_4])^* \mid p_1 \mid p_1 \mid p_4 \]
Finite State Automata

Assuming ONE automaton and ONE input string.

FSA to Strand Algebra

\[ ([A,a].[C,\tau])^* \mid ([A,b].[B,\tau])^* \mid ([B,c].[C,\tau])^* \mid ([C,d].[C,\tau])^* \mid ([C,d].[A,\tau])^* \mid A \mid \tau \]

Input strings

\[ a,b,c,d \]

Automata populations are a more natural model...
Interacting Automata

This is a uniform population of identical automata, but heterogeneous populations of interacting automata can be similarly handled.
This is a uniform population of identical automata, but heterogeneous populations of interacting automata can be similarly handled.
This is a uniform population of identical automata, but heterogeneous populations of interacting automata can be similarly handled.
This is a uniform population of identical automata, but heterogeneous populations of interacting automata can be similarly handled.
Strand Displacement
Intermediate Language
A. Syntax of DNA molecules $D$

Upper strand with sequence complementary to $S$

$S$ 
$<S>$

Molecule with segments $G_1, ..., G_k$

$G_1: G_2 : ... : G_k$

Parallel molecules $D_1, ..., D_k$

$D_1 | D_2 | ... | D_k$

B. Syntax of DNA segments $G$

Lower strand with toehold $N^o$

$N^o$

$N^c$

Double strand with sequence $S$ and overhangs $L, R$

$<L>[S]<R>$

C. Syntax of DNA sequences $S, L, R$

Sequence of domains $O_1, ..., O_k$

$O_1 O_2 ... O_k$

$O_1 O_2 ... O_k$
Dynamics

1. Toehold binding and unbinding

\[
\text{L} \quad \text{N} \quad \text{R} \quad \xrightarrow{-N_{c}} \quad G_{1} \quad N_{c} \quad G_{2} \quad \xleftarrow{+N_{c}} \quad G_{1} \quad G_{2}
\]

\[
\text{L} \quad \text{R} \quad N \quad \text{c} \quad G_{1} \quad N_{c} \quad G_{2}
\]

2. Strand displacement to the right

\[
\begin{array}{ccc}
\text{L}_{1} & \text{L}_{2} & \text{S}_{1} \\
\text{S}_{1} & \text{S}_{2} & \text{R}_{1} \\
\end{array} \quad \rightarrow \quad \begin{array}{ccc}
\text{L}_{1} & \text{S}_{1} & \text{S}_{2} \\
\text{R}_{1} & \text{S}_{2} & \text{R}_{2} \\
\end{array}
\]

\[
\text{L}_{1} \quad \text{S}_{1} \quad \text{S}_{2} \quad \text{R}_{1} \quad \text{S}_{2} \quad \text{R}_{2}
\]

3. Strand displacement to the left

\[
\begin{array}{ccc}
\text{L}_{1} & \text{L}_{2} & \text{S}_{1} \\
\text{S}_{1} & \text{S}_{2} & \text{R}_{1} \\
\end{array} \quad \rightarrow \quad \begin{array}{ccc}
\text{L}_{2} & \text{S}_{1} & \text{S}_{2} \\
\text{R}_{1} & \text{S}_{2} & \text{R}_{2} \\
\end{array}
\]

\[
\text{L}_{2} \quad \text{S}_{1} \quad \text{S}_{2} \quad \text{R}_{1} \quad \text{S}_{2} \quad \text{R}_{2}
\]

4. Branch migration

\[
\begin{array}{ccc}
\text{L}_{1} & \text{L}_{2} & \text{S}_{1} \\
\text{S}_{1} & \text{S}_{2} & \text{R}_{1} \\
\end{array} \quad \Rightarrow \quad \begin{array}{ccc}
\text{L}_{1} & \text{L}_{2} & \text{S}_{1} \\
\text{S}_{1} & \text{S}_{2} & \text{R}_{1} \\
\end{array}
\]

\[
\text{L}_{1} \quad \text{L}_{2} \quad \text{S}_{1} \quad \text{S}_{2} \quad \text{R}_{1} \quad \text{R}_{2}
\]
Strand Displacement Simulation Tool

1 Transducer gate \(x.y\) (3 initial species)

directive sample 30.0 1000
new xt@1.0,1.0
new yt@1.0,1.0
( 1000 *<xh xt^: xb>
1000 *xt^:[xb yt^-]<yb>:a[
1000 *<yt^ a>
)

\(a\) fresh; \(x_t\) generic

\(x | x.y \rightarrow y\)
Fork Chain Reaction $x.[x,x]$ (3 initial species)

directive sample 30.0 1000
directive plot "="<reporter>"
new xt@ 1.0 , 1.0
( 1 * <xb xt^:xb>
| 1000 * xt^:xb>:[xb]:[reporter]
| 1000 * <xt^ a xt^ reporter>
}

26 Species, 20 Reactions
Strand Displacement Simulation Tool

1 Join gate with garbage collection \([x,y].z\) (8 initial species)

34 Species, 18 Reactions

Luca Cardelli  2009-07-11  44
Sequence Design
Sequence Design

NUPACK BETA nucleic acid package

Input

Nucleic acid type: RNA DNA
Number of designs: 1
Target structure:

Preview:

Output

Designability summary

Sequence designs

Average percentage of correct nucleotides:

Average number of incorrect nucleotides:

GC content:

Sequence:

Copyright © 2007-2009 Caltech. All rights reserved. | Contact | Funding | Terms of use
Conclusions
Conclusion

- **Nucleic Acids**
  - Programmable matter

- **DNA Strand Displacement**
  - A computational mechanism at the molecular level

- **DNA Compilation**
  - High-level languages (Boolean Networks, Petri Nets, Interacting Automata)
  - Intermediate languages (Strand Algebra, Strand Displacement Language)
  - Sequence generation

- **Tools**
  - Thermodynamic analysis
  - Simulation
  - Verification (not yet)